Technical White Paper

Dynamic Multi-Protocol Manager (DMM) Performance

ABSTRACT

The TI Dynamic Multi-Protocol solution allows multiple wireless protocols to run concurrently using a single radio, handling any timing conflicts that occur when multiple protocols request access to the radio during the same time period. To understand the system performance ramifications of operating multiple protocols in parallel, it is important to understand how the scheduler is making decisions. Some key questions to look for on any multi-protocol solution are as follows:

- What happens when a timing conflict occurs and multiple protocols need the radio?
- How will concurrent operation affect end to end reliability and latency?
- How will the maximum throughput of a protocol be affected given timing conflicts and how can I optimize throughput performance?
- How can I configure the scheduler to match my use case?

The following sections cover these key questions specific to the TI DMM implementation that has been extensively tested and verified across various network configurations for IEEE 802.15.4 based protocols, Zigbee, and TI 15.4 operating concurrently with *Bluetooth*® Low Energy.

Table of Contents

Trademarks

1 Introduction

The SimpleLink™ platform from Texas Instruments offers the broadest portfolio of differentiated wired and wireless Arm® MCUs featuring Ethernet, *Bluetooth* Low Energy, Wi-Fi®, Sub-1 GHz, IEEE 802.15.4, Zigbee®, and Thread. They are all unified by a single, robust, and comprehensible SimpleLink CC13x2 and CC26x2 software development kit (SDK) with 100% application code portability, modular development kits, and cloud-based tools. The flexibility of the SimpleLink platform helps manufacturers quickly develop and seamlessly reuse resources to expand their portfolio of connected products.

Connectivity solutions continue to be widely adopted for sensing and monitoring in applications like Building Security and HVAC systems. A growing trend in these industries is to leverage multiple wireless technologies to enhance and expand applications by capitalizing on the unique benefits of different wireless standards (that is, BLE for smartphone connectivity, Zigbee® for mesh, Sub-1 GHz for long range and low power). The Texas Instruments™ SimpleLink™ platform has the hardware to support these applications along with a software module called the Dynamic Multi-protocol Manager (DMM).

Dynamic Multi-protocol Manager allows multiple wireless stacks to coexist and operate concurrently. It acts as an arbiter between multiple stacks and the shared RF core resource. Texas Instruments' single-chip multi-standard solution supported by the DMM is not only cost-effective, but also simplifies multiprotocol design. Without DMM, supporting two wireless protocols requires a two-chip design, which can increase board size and expenses for designers. Alternatively, a single-chip cannot operate two protocols at the same time, and must bootload a new firmware image every time it switches between them. With compatible SimpleLink devices, users can leverage a single wireless MCU and support multiple different wireless standards. Getting started with the DMM is further simplified through application examples provided in the SDK as well as SimpleLink Academy, which gives a comprehensive and interactive learning experience to cover all phases of development from basic examples to advanced topics.

2 Background

What happens when a timing conflict occurs and multiple protocols need the radio?

Assuming two stacks operating in parallel, timing conflicts when both protocols request access to radio at the same time are inevitable. To resolve the conflict, a method must be defined to indicate which protocol stack will receive access to the radio in that instance. While many concurrent multiprotocol solutions rely on fixed-priority implementations, where a single stack is always higher priority than the other, the TI DMM solution is highly customizable and lets the developer set priorities according to any state of the system. This customization includes the ability to dynamically change the priorities based on either the state of the application or the state of the protocol stack. Given this information, the following three scenarios can occur for any given conflict. Note that the commands shown in Figure 2-1 through Figure 2-3 generally represent a transmit and receive pair, combined for simplicity.

Rejected Commands

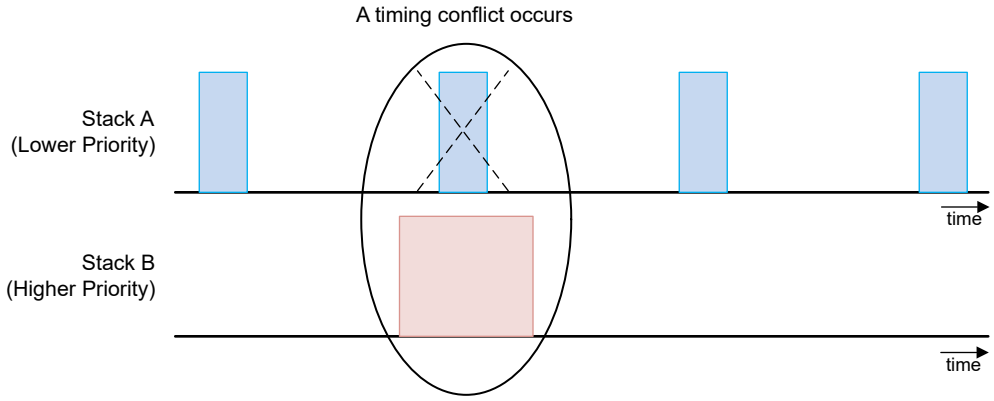

Figure 2-1. Rejected RF Packet

In Figure 2-1, stack A is sending packets on a short periodic interval, a typical use case for Bluetooth Low Energy where data is exchanged at a preconfigured connection interval, while stack B is sending periodic traffic

on a larger interval. On the second packet above, Stack B is already transmitting when stack A needs the radio, creating a timing conflict. Because stack B was scheduled with higher priority, it continues to transmit without interruption, and the stack A RF command is rejected. In the case of Bluetooth Low Energy, this means that any Bluetooth Low Energy data will be exchanged as part of the following connection event. If this were to occur to a protocol like Zigbee: MAC, network-, or application-level retries can be used to reschedule the packet. Many protocols feature retry logic as noted for Bluetooth and Zigbee to prevent data loss due to network instability. While this logic is not specific to the DMM use case, it can be used to ensure end-to-end reliability of data when timing conflicts occur.

Aborted Commands

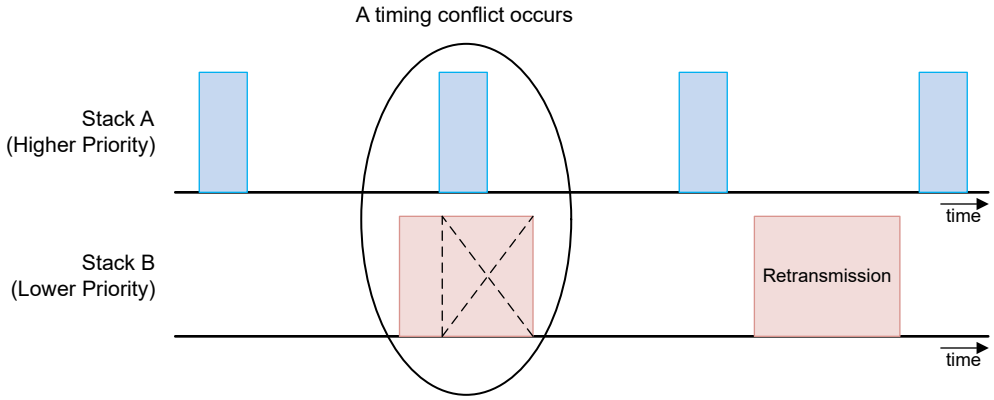

Figure 2-2. Aborted RF Packet

In Figure 2-2, the priority of the packets is switched between stack A and stack B compared to the previous image. This means that when the timing conflict occurs, the current transmission for stack B is lower priority than the scheduled command for stack A, causing the command for stack B to be aborted in favor of giving stack A radio access. Stack B then retransmits the failed packet, resulting in a successful transmission.

Delayed Commands

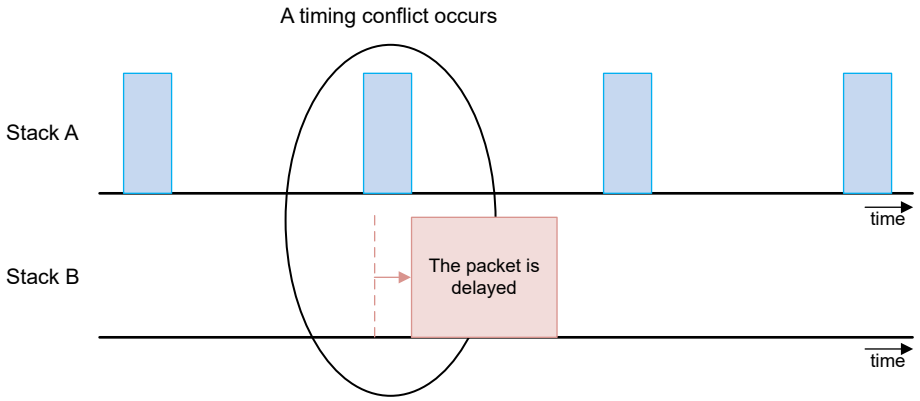

Figure 2-3. Delayable Packet

In Figure 2-3, both stack A and stack B request access to the radio during the same time period. Assume that stack A is a timing-critical packet, typical of the Bluetooth Low Energy use case, and that stack B is not timing-critical, meaning the packet can be delayed without consequences to the stack. In this scenario, the DMM scheduler can delay the stack B packet to avoid the timing conflict. One example of this use case is an asynchronous RF protocol where the collector always keeps its receiver on. In this case, a sensor node can send a packet at any time.

3 Verified for Reliability and Performance

To achieve a significant degree of confidence in the results shown in the following sections, a large quantity of data was collected and analyzed from multiple SimpleLink CC13x2 and CC26x2 SDKs released in 2019 and 2020. While the full data sample is not tested for each release, a subset of tests are performed to ensure congruence with the depicted results.

How will concurrent operation affect end-to-end reliability and latency?

The most common DMM use case encompasses Bluetooth running in parallel with an IEEE 802.15.4 based protocol. In this scenario, Bluetooth provides ease of use by supporting phone or tablet connections. Meanwhile the 802.15.4 network provides low-power long-range communication, and, in the case of Zigbee, mesh networking. While the TI DMM solution is not limited to these protocols, they have been used to verify system performance across a wide range of network configurations including but not limited to the following:

- Varying packet intervals
  - Allowable Bluetooth Low Energy connection intervals between 7.5 ms and 4000 ms ([7.5, 30, 50, 100, 130], [200-2000 in steps of 100], [2000-4000 in steps of 500])
  - IEEE 802.15.4 packet intervals ≥ 250 ms
- Varying number of connections
  - Note: The number of supportable Bluetooth Low Energy connections depends on the traffic profile of the solution. While TI DMM performance has been verified with up to 4 connections as shown in Figure 3-1, this is not a limit on the system but a benchmark for analysis.
- Varying data rates affecting transmission lengths
  - For the IEEE 802.15.4 network: 5, 50, and 200 kbps operating in Sub-1 GHz, and 250 kbps operating in the 2.4-GHz frequency band
  - All Bluetooth 5.0 PHY rates
- Multiple IEEE 802.15.4 modes of operation including non beacon (used by the Zigbee protocol stack), beacon, and frequency hopping (in Sub 1 GHz)

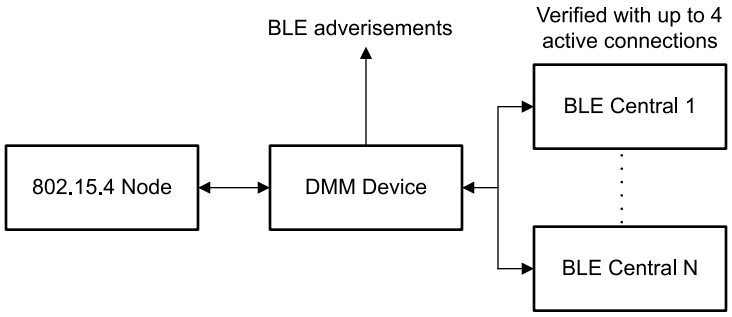

Figure 3-1. Dynamic Multi-Protocol Test Setup

As previously noted, timing conflicts are inevitable; however, the effect of these timing conflicts on system performance can be minimized by delaying packets that are not time critical and rescheduling unsuccessful packets during previously unused air time. In the typical use case, there is periodic traffic on both protocols within a multi-protocol system and, by the nature of low-power networking protocols, there will be a significant amount of time spent either sleeping to reduce power usage or, in the case of always-on devices such as routers within the low-power network, listening for packets. This time period that was previously unused can now be effectively repurposed to ensure end-to-end reliability within the system in the way of MAC, network layer, or application level retries. Many protocols such as Bluetooth Low Energy and Zigbee already features retry logic to prevent data loss due to network instability. This logic is used in the TI DMM system. The observed results show 100% end-to-end reliability for both Bluetooth Low Energy messages and the IEEE 802.15.4 based application traffic (see Figure 3-2).

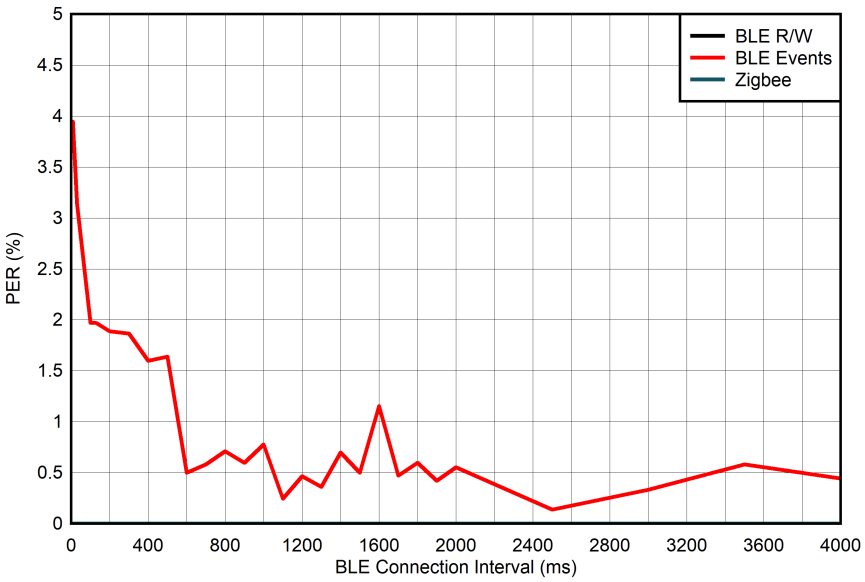

Figure 3-2. PER Performance of a DMM Zigbee End Device + Bluetooth Low Energy Peripheral

It is important to note while retries might solve potential packet losses when running a multi-protocol system, they will also introduce a (potential) end-to-end latency depending on how many times a packet has to be retried in order to be successful. This latency increase is the same for all packet loss reasons, it is not particular to using TI DMM. The potential increase for end to end latency will be minimal in most use cases and it is only expected to occur on the lower priority RF commands which can be configured to match the requirements of the design.

Figure 3-2 shows such a scenario. The setup consists of a TI DMM Zigbee switch + Bluetooth Peripheral, a Zigbee Light, and a Bluetooth Central. The TI DMM device is sending 48-byte light toggle to the Zigbee light 1 hop away on a 1-second periodic interval. In response to the light toggle from the switch, the light responded with a 45-byte application-level acknowledgement creating two way traffic. The TI DMM device is concurrently maintaining a Bluetooth connection with GATT read and writes occurring every 1.25 seconds from a Bluetooth central and advertisements are scheduled every 100 ms. Both MAC and network retries are enabled but application level retries are not. With consistent traffic on both Bluetooth and Zigbee, one might think that many timing conflicts occur causing a loss in data or increase in latency on the lower priority Bluetooth packets. In actuality:

- The end-to-end reliability was measured over a 30 minute period in an open-air environment to be 100% (0% PER) for both the Zigbee light toggles and the Bluetooth GATT messages.
- There was no increase in latency for the higher-priority light toggles when compared to a standalone Zigbee device.
- The increase in latency of the lower-priority Bluetooth reads and writes was negligible as the event error rate, measuring the ratio of timing conflicts on Bluetooth resulting in potential delayed messages, was on average approximately 1%.

Note that low levels of PER can still occur on some DMM examples depending on the number of retries selected or when operating under stressful network traffic conditions, such as a short packet intervals at a 5-kbps data rate on the IEEE 802.15.4 protocol resulting in larger transmission times.

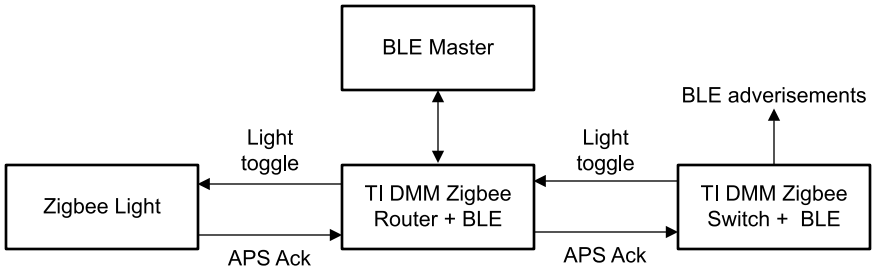

Figure 3-3. TI DMM Router PER Test Setup

The test setup in Figure 3-3 is a variation of that shown in Figure 3-1 made to verify the ability of a TI DMM device to route packets between two nodes in a mesh network. The network comprises a TI DMM Zigbee switch sending 48-byte light toggle messages to the Zigbee light on the network 2 hops away. The toggles were scheduled randomly every 250 to 500 ms and the device also executed concurrent Bluetooth Low Energy advertisements every 100 ms. In response to the light toggle message from the switch, the light responds with a 45-byte application-level acknowledgement creating two-way traffic. The TI DMM Zigbee router forwards messages between the two other nodes in the Zigbee network while concurrently maintaining a Bluetooth Low Energy connection with GATT read and writes occurring every 1.25 s from the Bluetooth Central. Both MAC and network retries are enabled but application-level retries are not. The end-to-end reliability was measured over a 30-minute period in an open-air environment and was 100% for both the Zigbee light toggles and the Bluetooth Low Energy GATT messages.

How will the maximum throughput for a stack be affected given timing conflicts and how can I optimize throughput performance?

Any multi-protocol solution has the inherent limitation that the time used by each individual stack within the system must be less than 100% utilization of the radio and that the protocol cannot have greater performance within the multi-protocol system than that observed in a single-protocol system. With this limitation, the throughput performance within the system can be estimated based on the the network traffic. Using a reduced version of the test setup as shown in Figure 3-1 in an open-air environment, one-to-one throughput performance was captured for Bluetooth Low Energy while periodic traffic was sent on the IEEE 802.15.4 network. To achieve maximum throughput, Bluetooth Low Energy advertisements were disabled after establishing a connection. The captured data was then compared to a derived model for estimating system performance based on the limitation that the radio utilization must add up to 100%. The results show the accuracy of the model, averaging less than 2% difference between the measured and estimated results regardless of the numerous changes in configuration as noted in the previous section. For more information on how to use this experimental model to estimate system performance, see the *System Performance and Limitations* section of the DMM user's guide.

Figure 3-4 uses the model to show the achievable throughput relative to the maximum achievable throughput in a single-protocol system with 100% performance equivalent to that observed in such a system. The expected throughput is mapped with respect to the Bluetooth connection interval, defined as the periodic interval between scheduled keep alive events. The black line refers to a static priority system where in this instance Bluetooth will always be higher priority than the 802.15.4 stack. The red line refers to the TI DMM solution utilizing dynamic priority to modify the priority of Bluetooth commands at run time based on the current state of the protocol stack. Note that the overhead caused by the TI DMM solution is negligible with respect to system performance and that instantaneous throughput for a given stack will not be affected within the DMM system for any time period during which a timing conflict does not occur.

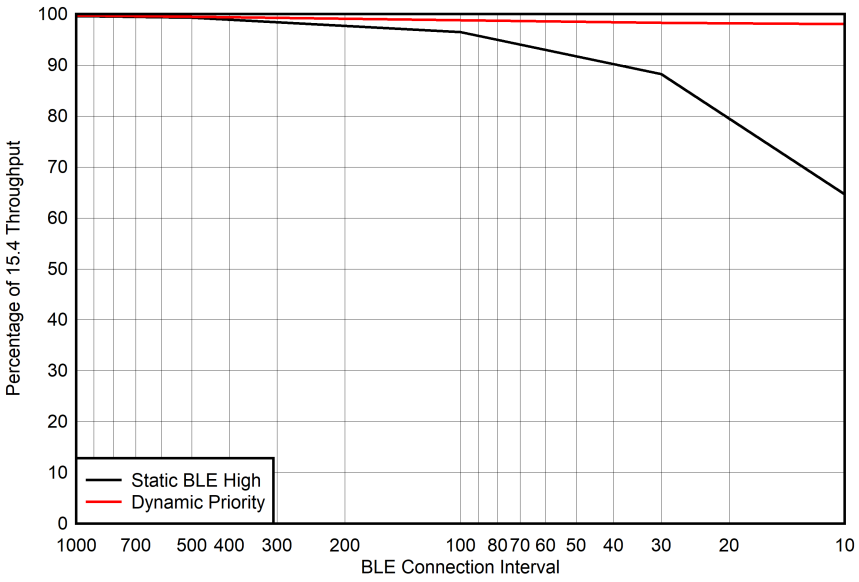

Figure 3-4. Throughput Benefits of Dynamic Priority

As noted earlier, many multi-protocol solutions have a static priority system in which one stack always wins out over the other. This can cause drastic ramifications on throughput performance within a multi-protocol solution. For example, consider a multi-protocol system with a Bluetooth Low Energy peripheral set to high priority in parallel with a Zigbee end device set to low priority. If the throughput is stressed on the Bluetooth Low Energy link, the Zigbee end device will be starved of the radio, resulting in an inability to send or receive data until the Bluetooth Low Energy traffic is completed. The TI DMM solution is able to avoid this issue in two ways: out of box priority settings and dynamic priority.

In the TI DMM Bluetooth Peripheral plus Zigbee end device out of box example, Bluetooth is set by default to a lower priority than Zigbee data. This means that while Bluetooth is stressing the radio bandwidth, Zigbee packets are still able to access the radio, and its operation on the network is maintained. However, wouldn't this just create an issue in the opposite direction when the Zigbee link is stressed with maximum throughput causing Bluetooth Low Energy to be starved? This is where dynamic priority comes in. To support high throughput, the Bluetooth Low Energy stack is able to dynamically increase the priority of its connection events after a given amount of time without a successful event. This prevents a supervision timeout, where a Bluetooth connection is terminated after a pre-negotiated amount of time without a successful event. This allows both the connection and end-to-end reliability to be maintained while also resulting in significantly improved throughput on the Zigbee link. Figure 3-4 shows the differences in throughput in this scenario between the TI DMM solution and a static-priority system. While throughput will decrease as the frequency of Bluetooth Low Energy events increases, the decrease is mitigated by dynamic priority to improve the overall system performance.

It is important to remember that the model for estimating throughput within the TI DMM solution was derived irrespective of the protocol stacks operating within given system. It was then proven using an IEEE 802.15.4 network operating in parallel with Bluetooth, showcasing both the accuracy of the model over approximately 800 data points in addition to proving that the performance within the TI DMM solution matches theoretical limitations. As a result, this model can be applied to any combination of protocol stacks to reliably predict expected system performance given a traffic profile.

For more information on how to estimate throughput within the TI DMM solution or DMM performance in general, see the *System Performance and Limitations* section of the DMM user's guide.

4 Optimizing the DMM Scheduler for Your Use Case

How can I configure the scheduler to match my use case?

TI releases out of box examples for Bluetooth acting in parallel with EasyLink, TI 15.4, and ZigBee. Each of these examples have been optimized for performance with Bluetooth serving as lower priority by default and using dynamic priority to increase the priority of stack-specific activities as needed. The reason Bluetooth was chosen to be lower priority by default is because Bluetooth is assumed to have a shorter periodic interval than the protocol that is running in parallel. This means that missing events for the concurrent protocol to operate has a less significant effect on performance and also minimizes the retries required on the protocol operating in parallel. That being said, the out of box examples may not match every use case, so TI's DMM solution provides customers extensive tools to take advantage of dynamic priority at both the application level and stack level. For more information on how to dynamically modify priorities for a given system, see the *Application States and Scheduling Policies* section of the DMM user's guide.

5 Recreating PER Results

As part of the quarterly release of the SimpleLink CC13x2 and CC26x2 SDK, DMM examples support project defines for measuring PER. To build these examples, follow these steps:

1. Build DMM Example Projects using PER project defines
   a. DMM TI 15.4 examples
      i. Create a workspace for the desired TI 15.4 projects: dmm_154collector_remote_display_app, dmm_154sensor_remote_display_app, collector, sensor, or corresponding 2.4Ghz project variants.
      ii. To change the traffic profile of the network, refer to the TI 15.4 sycfg module. Key configurable are "Mode", "Phy type", "Report Interval", "Polling interval", and "Tracking interval".
      iii. On either the DMM sensor or standalone sensor, define DISPLAY_PER_STATS.
   b. DMM Zigbee examples
      i. Create a workspace for the desired Zigbee projects: dmm_zed_switch_remote_display_app, dmm_zr_light_remote_display_app, and zc_light
      ii. For the DMM ZED add the following defines: PER_TEST, PER_TEST_SEND_USE_FINDING_BINDING, SAMPLEAPP_PER_TEST_SEND_CMD_DELAY and set SAMPLEAPP_PER_TEST_SEND_CMD_DELAY to desired zigbee light toggle interval in ms.
      iii. For the DMM ZC add the following defines: PER_TEST, PER_TEST_DISABLE_FINDING_BINDING, PER_TEST_ENABLE_FWD_NOTIFICATION
         1. For the ZC ensure the DMM zed cannot choose the ZC as it's parent by defining NWK_MIN_ROUTER_CHILDREN=NWK_MAX_DEVICES, ensuring the routing capabilities are being verified.
      iv. The number of MAC and Network level retries can be adjusted using the Zigbee syscfg module to modify configurable "MAC Frame Attempts" and "NWK Data Retries"
2. Flash images to TI LaunchPad™ kits with one of the following topologies.
   a. TI 15.4 Collector (DMM or standalone), TI 15.4 Sensor (DMM or standalone)
   b. ZC Light, DMM ZED Switch
   c. ZC Light, DMM ZR Light, DMM ZED Switch
3. Setup Network topology according to the README of the example. At this time, PER can be observed for the 802.15.4 network while BLE is in the advertising state. If monitoring performance for an active Bluetooth connection is desired, move on to step 4.
   a. For 15.4 Networks, observe PER measurement print to CUI status line "Sensor PER." The printed value represents the ratio of failed sensor messages, sent on a periodic interval configured by the "Report Interval" from step 1.a.i.
   b. For Zigbee Networks, observe PER metrics printed to the CUI status line "PER Test."
      i. ZED:
         1. Queued: Light Toggle send accepted by the network layer to be sent
         2. Success: Light Toggle for which we receive application level acknowledgement
         3. Failed: Light Toggle failed to Queue
         4. NoAck: Light Toggle for which we did not receive an APS acknowledgement

ii. ZR Metrics:
    1. Queued: Forwarded packet send accepted by the network layer to be sent
    2. Success: Forwarded Packet for which we receive MAC acknowledgement
    3. NoRoute: Network layer says no route to destination
    4. NoMAck: No MAC acknowledgement received for forwarded packet
    5. Expired: Packet expired at MAC layer
iii. ZC: If the network is setup correctly, you should see the Red LED being toggled on the preconfigured interval.

4. Flash an additional TI launchpad with a Bluetooth host test image. Prebuilt images are provided as part of the SDK in the "ble5stack/hexfiles" directory for the desired launchpad.
5. For the results shown in this paper, internal automated scripts were used to create consistent Bluetooth traffic and monitor performance measurements. This process can be recreated manually using BTool to control a BLE central device. Refer to the BTool User Guide for instructions on how to form a Bluetooth connection, set the desired traffic profile, Read / Write characteristics, and monitor PER.
    a. After the connection is established and traffic profile set, the PER metrics can be read or reset using the following advanced command: "HCIEXT_PacketErrorRateCmd"

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (https:www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265